PROJET AVANCE EN SYSTEMES EMBARQUES

DEPLOIEMENT D'UN MOTEUR PEER TO PEER EMBARQUE SUR ROUTEUR IP AVEC GESTION D'UN ESPACE DE STOCKAGE

BRESSON Emeline
CHEVRIER Julien
CHRETIEN Sébastien
SEMELLE Loïc

ENSEIRB 2008/2009

Sommaire

Introduction 5

I. Présentation 5

II. Solution : Généralités 6

1. Le réseau Peer To Peer : BitTorrent 6

2. Solution : les routeurs WiFi 7

3. Solution : le serveur de médias 8

III. Solution détaillée 9

1. La distribution d'OpenWRT : Kamikaze 9

2. Le client BitTorrent : rTorrent 9

3. Le tracker : XBNBT 10

4. Le serveur FTP : VSFTP 12

5. Les scripts 12

i. Script des routeurs 12

ii. Script du serveur de médias 14

iii. Tâches Cron 16

6. wTorrent 16

i. Présentation 17

ii. XMLRPC 18

iii. Smarty 18

iv. Prérequis 19

v. Modifications 19

vi. Utilisation 22

7. La compilation croisée 23

i. Libsigc++ 23

ii. Libtorrent 23

iii. Xmlrpc 24

iv. rTtorrent 24

v. Libxml 25

vi. PHP 25

vii. Lighttpd 26

viii. Ncftp 26

Conclusion 27

Annexes 28
I. Mode d'emploi 28
1. Lancement du Media Server (ordinateur) 28
2. Lancement du routeur 28
3. Lancer un téléchargement 28
4. Rapatrier un fichier sur l'ordinateur 29
5. Administrer le Media Server 29
i. Ajout de fichiers 29
ii. Administrer XBNBT 29
II. Sources 30
1. Source ajoutées à wTorrent 30
i. Liste des fichiers disponibles sur le Media Server 30
ii. Affichage des fichiers locaux sur le routeur 32
2. Scripts de compilation croisée 33
i. Compilation croisée de xmlrpc-c 33
ii. Compilation croisée de rTorrent 34
iii. Compilation croisée de PHP 36
iv. Compilation croisée de Ncftp 36
v. Compilation croisée de Lighttpd 37
vi. Compilation croisée de Libxml 37
vii. Compilation croisée de Libtorrent 38
viii. Compilation croisée de Libsigc++ 38
ix. Compilation croisée de Curl 39

Table des illustrations

Figure 1: Architecture du réseau 5
Figure 2: Outils présents sur le routeur 7
Figure 3: Outils présents sur le serveur de médias 8
Figure 4 : Tracker XBNBT 11
Figure 5: Algorithme du script routeur 13
Figure 6: Algorithme du script blackbox 15
Figure 7: Interface de wTorrent 17
Figure 8: Chargement d'un fichier torrent 18
Figure 9: Séquencement des opérations suite à une requête utilisateur 20
Figure 10: Listes des fichiers disponibles sur le tracker 21
Figure 11 : Liste des fichiers locaux 22

Introduction

Actuellement, les réseaux haut débit sont en plein essor. Ceci a contribué au développement de réseaux P2P. De même les Box multifonctions du type Livebox ou Freebox sont devenues monnaie courante chez les particuliers. Au vu de cette évolution, des développeurs ont porté des distributions Linux dans l'idée de les embarquer dans des Routeurs Wifi pour remplacer l'OS par défaut. L'implémentation de Linux dans ce type de produit permet d'ajouter de nouvelles fonctionnalités.

Dans ce contexte, l'objectif du projet est de déployer un réseau P2P en prenant comme clients des routeurs. Ainsi les routeurs doivent pouvoir se connecter de façon autonome à un réseau P2P et télécharger des médias disponibles sur un serveur ou bien disponibles sur les autres box.

I. Présentation

Le projet à réaliser est composé de deux types d'éléments différents

- un serveur de média mettant à disponibilité plusieurs fichiers vidéo
- des routeurs sans-fil simulant des Box ADSL

Tous ces éléments communiqueront sur un réseau Ethernet pour tester leur fonctionnalité. Ceux-ci seront reliés selon le schéma suivant:

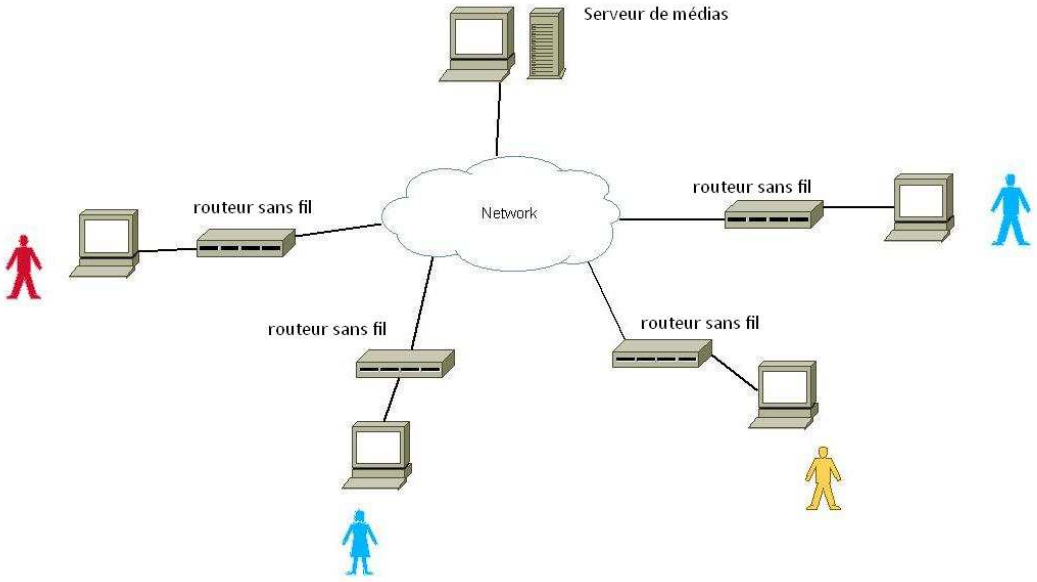

Figure 1: Architecture du réseau

Chaque routeur doit posséder son propre espace de stockage pour pouvoir télécharger des fichiers vidéo depuis un serveur de médias, ou directement enregistré depuis une plateforme MythTV reliée au routeur.

Les routeurs doivent également posséder une interface de contrôle simple, pour que tout utilisateur lambda puisse démarrer le téléchargement d'un fichier vidéo d'un simple clic.

Le serveur de média doit pouvoir mettre à disposition plusieurs vidéos. De plus, il doit pouvoir recevoir des vidéos depuis un des routeurs. Il y a peu de contraintes concernant ce serveur: Cela peut être un simple ordinateur de bureau, un serveur ou un routeur.

Afin de simplifier les protocoles d'échanges de fichiers, le réseau BitTorrent sera utilisé.

II. Solution : Généralités

La solution définie comporte trois parties principales. Tout d'abord, le choix du réseau peer to peer qui allait être utilisé a représenté une étape importante, ainsi que celui des clients et tracker associés. Les deux autres parties concernent d'une part toutes les opérations à réaliser sur les routeurs sans fil, et d'autre part sur le serveur de médias.

Ces deux dernières parties sont expliquées brièvement par la suite. Le détail est donné dans la partie concernant la solution détaillée.

1. Le réseau Peer To Peer : BitTorrent

Le but du projet étant de mettre en place un réseau peer to peer, il a fallu choisir un protocole de transfert de données.

Le choix s'est porté sur le protocole BitTorrent, qui présente de nombreux avantages par rapport à des concurrents comme Emule, Gnutella ou encore KaZaA. Ce protocole ne mise pas sur la longévité de disponibilité des données ou sur la diversité des sources. Il vise par contre une optimisation de la bande passante, aussi bien en envoi qu'en réception, ainsi qu'un débit maximal de flux en continu. Les fichiers sont découpés afin d'être partagés. Ainsi, des fichiers de plusieurs Go peuvent être partagés, et ce très rapidement.

Le protocole BitTorrent est parti du constat suivant :

- Quand une information se trouve sur un serveur unique, plus elle est demandée, moins elle est accessible à cause de la saturation du serveur.

Ainsi, BitTorrent renverse cette tendance en permettant à chaque client ayant téléchargé ne serait-ce qu'une toute petite partie de l'information, de la partager à son tour. L'information est donc découpée en petits segments distribués à des clients différents pour qu'ils aient tous quelque chose à échanger.

L'efficacité du réseau est donc maximale lorsqu'il y a beaucoup d'utilisateurs, car tous ceux qui téléchargent partagent par construction ce qu'ils téléchargent. Il n'y a pas à attendre dans une file virtuelle pour commencer.

Dans le langage BitTorrent, une personne possédant un fichier en totalité est appelé un seed (graine). Une personne pouvant transmettre une partie de l'information sans toutefois la posséder en entier est appelée un peer.

Pour fonctionner, le réseau BitTorrent nécessite un tracker. C'est un serveur qui sait en permanence quels seeds et quels peers se trouvent dans le swarm (l'essaim). Les clients lui envoient régulièrement

des informations. En échange, ils reçoivent des informations sur d'autres clients auxquels ils peuvent se connecter. Le tracker n'est pas directement impliqué dans le transfert de données et ne possède pas de copie du fichier.

De même pour utiliser ce protocole, les utilisateurs doivent posséder un client BitTorrent pour télécharger des torrents, en créer et les partager.

2. Solution : les routeurs WiFi

Le routeur que nous avons utilisé est WL-500g Premium de chez Asus. Il comporte les caractéristiques suivantes :

Mémoire RAM :	32 Mo
Mémoire FLASH :	8 Mo
Chipset Wifi :	Broadcom 4318
Chipset Ethernet :	Broadcom BCM5325
USB :	2 x USB 2.0
Processeur :	MIPS endianness little (MIPSel)

© 2006 AsusTek Computer Inc. All Rights Reserved.

Les identifiants du routeur sont les suivants :

Login : root

Mot de passe : root

Sur les routeurs devront être installés quelques outils nécessaires à la réalisation de la solution envisagée. Ils sont résumés dans le schéma suivant. Chacun d'eux sera ensuite détaillé dans la partie suivante du rapport.

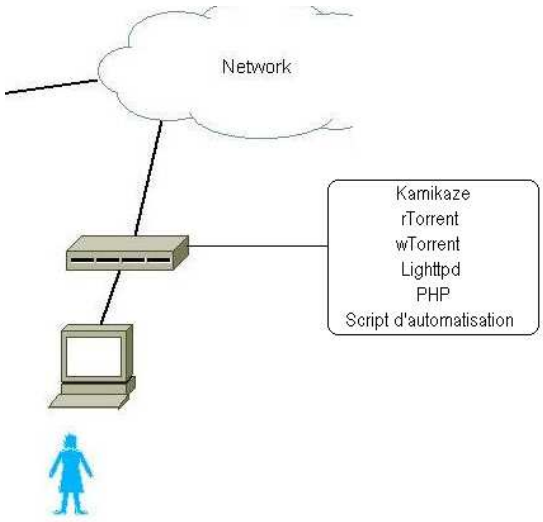

Figure 2: Outils présents sur le routeur

Kamikaze correspond à la distribution Linux installée sur le routeur. Le fait d'avoir installé Linux permet d'accéder à toute une panoplie de projets libres.

rTorrent est le client BitTorrent que nous avons sélectionné pour être embarqué dans le routeur.

wTorrent a été choisi pour piloter rTorrent. Il s'agit d'une interface web.

LightTpd est un serveur web. Il est utilisé pour deux fonctions :

- D'une part, il sert de passerelle de communication entre wTorrent et rTorrent
- D'autre part, il permet d'héberger wTorrent.

Le **module PHP** a été nécessaire car wTorrent est écrit en langage PHP.

Le **script d'automatisation** permet de vérifier si un utilisateur a ajouté un fichier sur la borne. Dans ce cas, il fait le nécessaire pour que le tracker référence ce nouveau fichier.

3. Solution : le serveur de médias

Pour réaliser le serveur de notre réseau, nous utiliserons un PC à notre disposition possédant les ressources suivantes : un CPU Via 1Ghz, 1 Go de mémoire et 80 Go de disque dur.

Nous avons installé la distribution Linux Fedora 10 à l'aide d'une clé live USB que nous avons créée car le pc ne dispose pas de lecteur CD.

Les identifiants de connexion pour ouvrir une session Fedora sont :

Login : loutre

Mot de passe : loutre

Tout comme dans les routeurs vus précédemment, plusieurs outils doivent également être installés sur l'ordinateur réalisant le serveur du réseau. Ils sont résumés dans la figure suivante et seront également détaillés dans la partie suivante. Certains sont les mêmes que dans les routeurs.

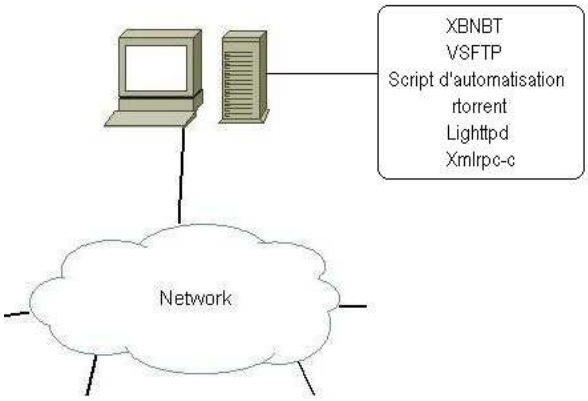

Figure 3: Outils présents sur le serveur de médias

Le Tracker **XBNXT** a été installé. Il s'occupe de mettre en relation les clients torrent et de répertorier les médias disponibles.

Grâce au serveur ftp **VSFTP**, les clients ont la possibilité d'envoyer au Media Server des fichiers .torrent sur le Media Server pour qu'ils soient disponibles auprès de tous les clients.

Le Media Serveur héberge différents types de fichiers : audio, vidéo... L'administrateur du serveur a la possibilité de rajouter des fichiers. Comme sur les routeurs, le **script d'automatisation** permet d'automatiser les tâches de création du .torrent associé et du référencement auprès du tracker.

Pour que le Media Server puisse partager les fichiers qu'il héberge, il est nécessaire d'installer un client torrent. **rTorrent** s'occupe de cette tâche.

Pour que le script d'automatisation puisse envoyer des commandes à rTorrent, il a été nécessaire d'installer un serveur web qui fait office de passerelle. C'est le rôle de **Lighttpd**.

III. Solution détaillée

1. La distribution d'OpenWRT : Kamikaze

Deux distributions différentes sont disponibles pour le firmware OpenWRT, Kamikaze et White Russian. Un petit comparatif a donc été établi dans le but de choisir. Il s'est avéré rapidement que Kamikaze possédait de nombreux avantages par rapport à son concurrent. En effet, White Russian ne supporte qu'un nombre très limité d'architectures, à savoir les équipements à base de Broadcom BCM47xx comme dans les routeurs Linksys WRT54G ou les Asus WL-500g. Ainsi, le pilote Wifi utilisé est le pilote propriétaire BroadCom. La configuration se fait par NVRAM (Non Volatile Random Access Memory) ce qui est spécifique à BroadCom. Au contraire, Kamikaze supporte un nombre important d'architectures, en plus des deux supportées par White Russian. Ainsi, il y a une abstraction de la configuration WiFi et de la gestion du réseau. Enfin, la configuration se fait grâce aux fichiers de configurations présents dans /etc/config. Ils sont ensuite interprétés par l'utilitaire UCI.

2. Le client BitTorrent : rTorrent

Il est nécessaire de posséder un client BitTorrent pour pouvoir utiliser le protocole de partage BitTorrent. Le client officiel porte le même nom que le protocole. Cependant, comme ce dernier est un protocole libre, de nombreux autres clients alternatifs ont été développés. Parmi les plus populaires, nous pouvons citer µTorrent, Azureus, ABC ou encore BitTornado.

Cependant, comme ce client est destiné à être installé sur des routeurs, possédant de faibles ressources, et à être utilisé à distance, il ne devait pas être choisi au hasard.

Le choix s'est porté sur rTorrent. C'est un client BitTorrent léger en mode console, écrit en C++ et dont l'interface est construite avec la bibliothèque ncurses. Il est de plus adapté pour être utilisé avec des outils comme screen ou dtach. L'interface de rTorrent est très légère et ne surcharge pas l'activité du processeur contrairement à certains clients BitTorrent lourds. Enfin, un simple fichier texte .rtorrent.rc (placé dans le répertoire personnel de l'utilisateur) permet de configurer tous les paramètres du client.

Outre les propriétés précédentes, indispensables à la réalisation de ce projet, rTorrent possède quelques caractéristiques intéressantes pour un réseau Peer to Peer. En effet, il possède quelques propriétés telles que le super seeding, le DHT ou le peer exchange.

DHT signifie table de hachage distribuée. C'est une technologie qui permet l'identification et l'obtention d'une information dans un système réparti. L'ensemble de la table de hachage est constitué virtuellement par tous ces constituants répartis sur tous les éléments du réseau, qui en possèdent chacun une partie.

Le super seeding est une fonction qui minimise le nombre de données que doit envoyer la source originale jusqu'à ce qu'un peer ait téléchargé le fichier complet. Cette fonction n'est utilisée que lorsqu'il n'y a qu'une seule source de téléchargement. Ainsi, l'unique source, au lieu de dire qu'elle possède le fichier complet, fait croire qu'elle n'en possède aucune partie. Dès que des peers se connectent la source en informe une qu'elle a reçu une partie du fichier, partie envoyée à aucun autre pair et l'autorise à la télécharger. La source ne va pas donner d'autre partie du fichier au même peer tant qu'elle n'aura pas reçu confirmation que cette partie a été renvoyée à nouveau. Le trafic au début d'un échange est donc facilité.

Le seul inconvénient de rTorrent est qu'il ne peut pas être utilisé pour créer des torrents. Cependant, ses caractéristiques étant très intéressantes, nous avons décidé de l'utiliser pour toutes les autres actions.

Un autre client BitTorrent nommé Ctorrent sera utilisé dans le seul but de créer les torrents. Très léger également, mais possédant moins d'avantages que rTorrent, il pourra sans problème être installé sur les routeurs en parallèle avec rTorrent.

3. Le tracker : XBNBT

Le tracker BitTorent permet de savoir qui télécharge quel torrent et où se trouvent les fichiers. C'est donc lui qui s'occupe de télécharger les torrents entre eux.

Nous avons choisi d'utiliser comme tracker XBNBT. Il est basé sur trois trackers différents : BNBT, Trinity EasyTracker et CBTT. Il reprend donc les avantages des trois comme le support des bases de données MySQL et dispose d'une interface graphique qui permet de gérer et télécharger les torrents.

De plus, XBNBT est codée en C++ pour une meilleure efficacité.

Pour installer XBNBT nous avons utilisé un package présent dans les dépôts de Fedora. L'installation se fait dans le dossier /usr/local/xbnbt.

Un fichier bnbt.cfg permet de configurer toutes les options de XBNBT.

Des scripts déjà réalisés permettent de lancer et d'arrêter facilement le tracker en utilisant les commandes suivantes :

- **xbnbt start** qui permet de démarrer le tracker
- **xbnbt stop** qui permet d'arrêter le tracker
- **xbnbt restart** qui permet de relancer le tracker

- **xbnbt booton** qui permet de lancer tracker au démarrage du system

Une fois celui-ci lancé nous accédons au tracker avec un navigateur internet en rentrant l'adresse suivante :

http://adresseIP:6969

Les identifiants de connexion pour administrer le tracker sont :

- Login : loutre
- Mot de passe : loutre

L'interface graphique est la suivante :

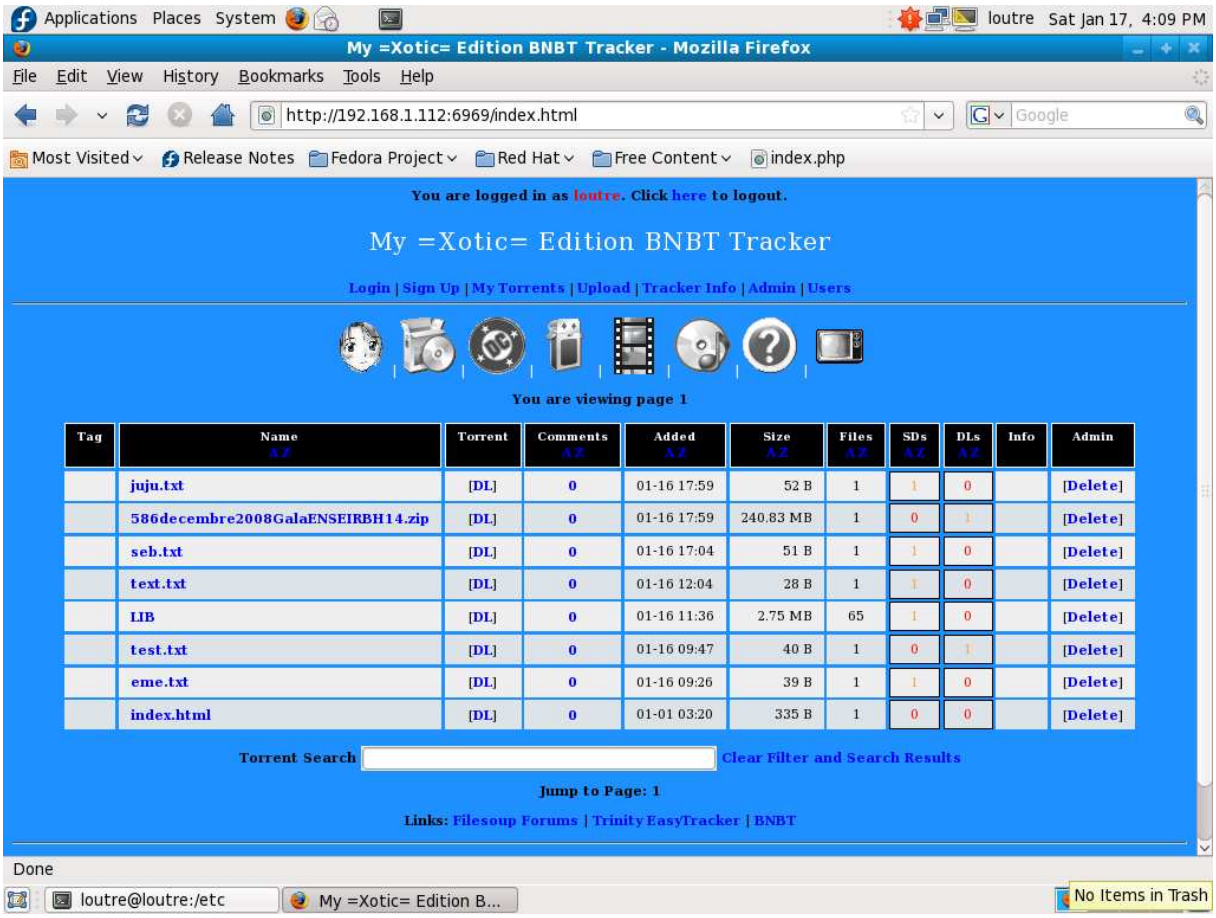
Applications Places System
loutre Sat Jan 17, 4:09 PM
My =Xotic= Edition BNBT Tracker - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.112:6969/index.html
Most Visited Release Notes Fedora Project Red Hat Free Content index.php
You are logged in as loutre. Click here to logout.
My =Xotic= Edition BNBT Tracker
Login | Sign Up | My Torrents | Upload | Tracker Info | Admin | Users
You are viewing page 1
Tag Name Torrent Comments Added Size Files SDs DLs Info Admin
juju.txt [DL] 0 01-16 17:59 52 B 1 1 0 [Delete]
586decembre2008GalaENSEIRBH14.zip [DL] 0 01-16 17:59 240.83 MB 1 0 1 [Delete]
seb.txt [DL] 0 01-16 17:04 51 B 1 1 0 [Delete]
text.txt [DL] 0 01-16 12:04 28 B 1 1 0 [Delete]
LIB [DL] 0 01-16 11:36 2.75 MB 65 1 0 [Delete]
test.txt [DL] 0 01-16 09:47 40 B 1 0 1 [Delete]
eme.txt [DL] 0 01-16 09:26 39 B 1 1 0 [Delete]
index.html [DL] 0 01-01 03:20 335 B 1 0 0 [Delete]
Torrent Search
Clear Filter and Search Results
Jump to Page: 1
Links: Filesoup Forums | Trinity EasyTracker | BNBT
Done
loutre@loutre:/etc
My =Xotic= Edition B...
No Items in Trash

Figure 4 : Tracker XBNBT

Elle permet de connaître le nom du fichier, la date d'ajout, la taille, le nombre de fichiers, le nombre de seeders et le nombre de leechers.

Elle permet aussi de télécharger le fichier .torrent qui permet de télécharger le fichier voulu et de les supprimer.

Les fichiers .torrent contiennent le hash du fichier, des informations sur le fichier et l'adresse du tracker.

4. Le serveur FTP : VSFTP

Nous utilisons un serveur FTP pour stocker les fichiers torrents. Le serveur ftp utilisé est VSFTP qui est un serveur ftp très sécurisé. L'installation se fait à partir des dépôts de Fedora.

A partir des mêmes dépôts nous trouvons un utilitaire de configuration avec interface graphique pour faciliter la configuration du serveur.

Ce serveur FTP pointe sur le dossier contenant les torrents utilisés par XBNBT. Ainsi à chaque fois qu'un fichier torrent est ajouté dans ce dossier, le tracker l'ajoute automatiquement à sa liste.

Pour se connecter au serveur FTP, il faut rentrer sont adresse IP et sélectionner le port 21.

Les identifiants de connexion sont les suivants :

- Login : loutre
- Mot de passe : loutre

5. Les scripts

Deux scripts ont dû être écrits pour faciliter la recherche de nouveaux fichiers multimédias ainsi que l'ajout des torrents correspondant sur le tracker. Le premier script est installé sur tous les routeurs, tandis que le second concerne le serveur. Dans un souci d'automatisation totale, et pour que l'utilisateur ne doive pas exécuter ces scripts de lui-même, ils seront de type « Cron » c'est-à-dire qu'ils seront exécutés périodiquement et automatiquement sans intervention de l'utilisateur.

i. Script des routeurs

Ce script devra d'une part repérer dans le dossier contenant les fichiers à partager s'il y a des nouveaux médias. Cela s'effectuera en regardant si chaque média possède un torrent qui lui est associé. Pour chaque nouveau fichier, le torrent associé sera créé, puis envoyé sur le serveur ftp. Enfin, rtorrent sera lancé pour que le tracker prenne en compte le nouveau fichier. D'autre part, pour chaque fichier .torrent déjà existant, sa présence sur le serveur ftp sera vérifiée. Dans le cas où le fichier n'y est pas, il sera envoyé, et rtorrent sera lancé pour que le tracker le prenne également en compte.

L'algorithme auquel répond ce script est le suivant :

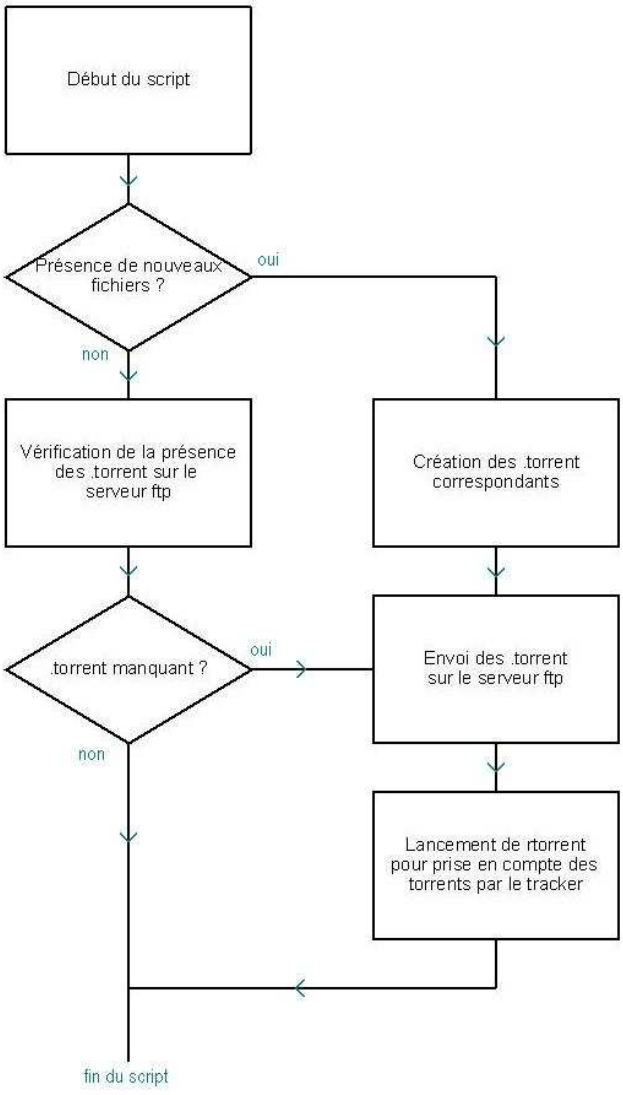
Début du script
Présence de nouveaux fichiers ?
oui
non
Vérification de la présence des .torrent sur le serveur ftp
Création des .torrent correspondants
.torrent manquant ?
oui
non
Envoi des .torrent sur le serveur ftp
Lancement de rtorrent pour prise en compte des torrents par le tracker
fin du script

Figure 5: Algorithme du script routeur

Le script peut donc être écrit en respectant cet algorithme. Le code écrit est le suivant :

```
#!/bin/sh

HOST='192.168.1.112'
PORT='6969'
USER='loutre'
PASSWD='loutre'
local_directory='/media/cle/wTorrent/Data'
if [ ! -s $local_directory ] (1)
  then exit
  else
    tri=$(ls $local_directory | grep -v ".torrent")
    cd $local_directory
    for i in $tri
      do
        coupe=$(echo $i | cut -d. -f1)

       if [ -s $coupe.torrent ] (2)
         then resuls=$(ncftpls -u $USER -p $PASSWD ftp://$HOST)
         if ! echo $resuls | grep $coupe.torrent (3)
```

```
                then ncftpput -u $USER -p $PASSWD $HOST . $coupe.torrent
        fi

      else
        ctorrent -t $i -u http://$HOST:$PORT/announce -s $coupe.torrent
        ncftpput -u $USER -p $PASSWD $HOST . $coupe.torrent
        xmlrpc localhost load_start $local_directory/$coupe.torrent

      fi
    done
fi
```

Les variables HOST et PORT correspondent respectivement à l'adresse IP et au port du tracker. Les champs USER et PASSWD correspondent au login et au mot de passe du serveur. En effet, ceux-ci sont requis pour l'accès au serveur ftp. Enfin, local_directory est l'adresse absolue du dossier des routeurs où sont stockés les médias à télécharger. Des variables ont été créées dans un souci de portabilité et de compréhension. Ainsi, si ces données changent, elles peuvent être facilement modifiées, une unique fois pour tout le programme.

La première condition (1) permet de vérifier si le dossier de stockage existe bien. La seconde (2) permet de vérifier si chaque fichier possède un torrent qui lui est associé. Enfin, la dernière (3) permet de voir si le torrent déjà existant est présent sur le serveur ftp.

Les commandes ncftpls et ncftpput permettent respectivement d'obtenir le contenu du serveur ftp et de transférer un fichier sur le serveur. Elles ont été utilisées pour leur simplicité. En effet, ces commandes sont directement en langage bash et leur résultat peut directement être stocké dans des variables. Cela évite d'avoir à écrire des commandes ftp à l'intérieur du script bash, d'autant plus que la récupération des résultats ne serait pas aisée. Ces commandes devront donc nécessairement être installées sur les routeurs.

Les torrents sont créés avec le logiciel ctorrent pour les raisons citées précédemment dans ce document. Pour que le script fonctionne, il a été défini une convention selon laquelle un fichier nom.xxx serait associé à un torrent nom.torrent. Cela n'est en effet pas obligatoire que les deux fichiers portent le même nom.

Enfin, la commande xmlrpc permet par l'intermédiaire du serveur http lighttpd d'envoyer une commande au logiciel rTorrent sans avoir besoin de relancer le logiciel dans une console. Ainsi, le fichier est chargé et le tracker prend bien en compte la « graine » qui vient d'apparaître.

ii. Script du serveur de médias

Le script porté sur le serveur de médias est globalement plus simple que celui porté sur les routeurs. En effet, le dossier contenant les fichiers du serveur ftp est le même que celui où le tracker répertorie les torrents existants. Ainsi, il n'est nul besoin d'envoyer les torrents créés par ftp ou même de les déplacer dans un dossier différent. Ce script se contentera donc de regarder si de nouveaux médias ont été placés dans le dossier. Dans ce cas, pour chaque nouveau média, le fichier .torrent associé sera créé, avec la même convention de nom que précédemment, et rTorrent sera lancé pour prendre en compte par le tracker.

L'algorithme décrivant ce comportement est le suivant :

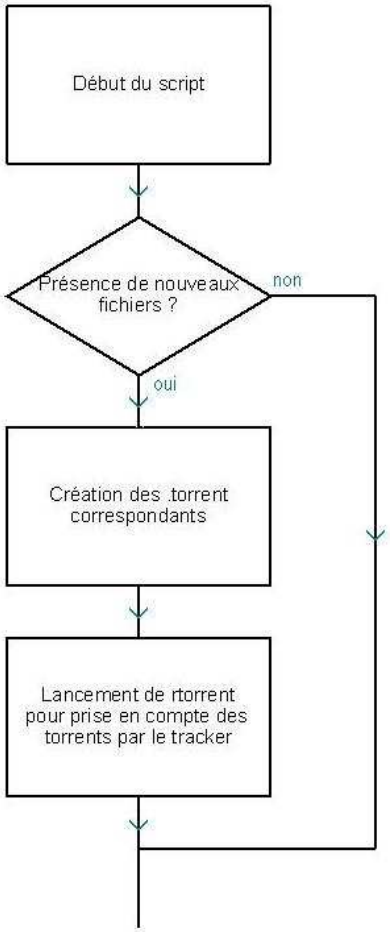
Début du script
Présence de nouveaux fichiers ?
non
oui
Création des .torrent correspondants
Lancement de rtorrent pour prise en compte des torrents par le tracker

Figure 6: Algorithme du script blackbox

Le script est donc écrit selon cet algorithme :

```
#!/bin/sh

HOST='192.168.1.112'
PORT='6969'
local_directory='/usr/local/bin/xbnbt/files/torrents'

if [ ! -s $local_directory ]
  then    exit
  else

    tri=$(ls $local_directory | grep -v ".torrent")
    cd $local_directory
    for i in $tri
    do
      coupe=$(echo $i | cut -d. -f1)

      if [ ! -s $coupe.torrent ]
      then
        ctorrent -t $i -u http://$HOST:$PORT/announce -s $coupe.torrent
        xmlrpc localhost load_start $local_directory/$coupe.torrent
      fi
    done
fi
```

Ce script est globalement plus simple à comprendre que le précédent. En effet, les mêmes commandes sont utilisées, et d'autres ont tout simplement été enlevées. Les variables du début sont les mêmes que précédemment. Elles ont la même fonction, et peuvent être modifiées selon les paramètres de la machine utilisée.

Après avoir vérifié que le dossier contenant les médias existe bien, il s'agit de vérifier que chaque fichier possède bien un torrent qui lui est associé. Si ça n'est pas le cas, le torrent est créé comme précédemment avec cTorrent, puis rTorrent est lancé en utilisant Xmlrpc.

iii. Tâches Cron

Ces deux scripts devront être exécutés périodiquement pour vérifier l'ajout de nouveaux fichiers à partager, et cela sans intervention des différents utilisateurs. Le principe d'utiliser une tâche « cron » répond tout à fait à la problématique. En effet, crontab est le nom du programme sous Linux qui permet d'éditer des tables de configuration du programme cron, c'est-à-dire la liste des tâches qui doivent être lancées à horaire fixe.

La table de configuration des tâches périodiques est située dans /etc. Il suffit donc d'éditer le fichier /etc/crontab pour y ajouter l'information sur le script à exécuter. Cela se présente de la façon suivante :

```
*/10 * * * * user /usr/local/bin/xbnbt/script_eirbox > >/dev/null
```

Chacun des 5 premiers groupes est en rapport avec une unité de temps, la première les minutes, puis les heures, le jour du mois, les mois et enfin le numéro du jour de la semaine (0 pour dimanche, 1 pour lundi…) Dans le cas présent, */x signifie toute les x unités de temps, donc la tâche sera exécutée toutes les 10 minutes. L'étoile signifie pour n'importe quelle unité de temps. Il faut ensuite préciser le nom d'utilisateur, ainsi que le chemin de la tâche à exécuter, ici le chemin donnant accès au script. Enfin, il faut préciser un fichier dans lequel sera stocké le journal des opérations. >/dev/null permet de s'affranchir de la journalisation.

6. wTorrent

Plusieurs interfaces existent pour pouvoir piloter rTorrent. Plusieurs implémentations sont disponibles : java (projet *ntorrent*), ncurses (interface par défaut) et web (projets *rtwi, wTorrent*...)

Parmi toutes ces implémentations, nous avons choisi d'utiliser une interface web pour deux raisons :

- D'une part, cela permet un pilotage à distance.
- D'autre part, une interface web est simple d'utilisation. En effet un simple navigateur web suffit.

Divers projets basés sur des interfaces web existent pour pouvoir piloter rTorrent : *rTwi, gi-torrent, wtorrent*... Nous nous sommes tournés vers le projet *wTorrent*. Les raisons de ce choix sont expliquées dans la partie suivante.

i. Présentation

wTorrent est une interface Web écrite en PHP pour rTorrent. Parmi tous les projets web existant*,* *wTorrent* sort du lot pour plusieurs raisons.

D'une part *wTorrent* fournit une interface intuitive et agréable comme le montre l'image suivante :

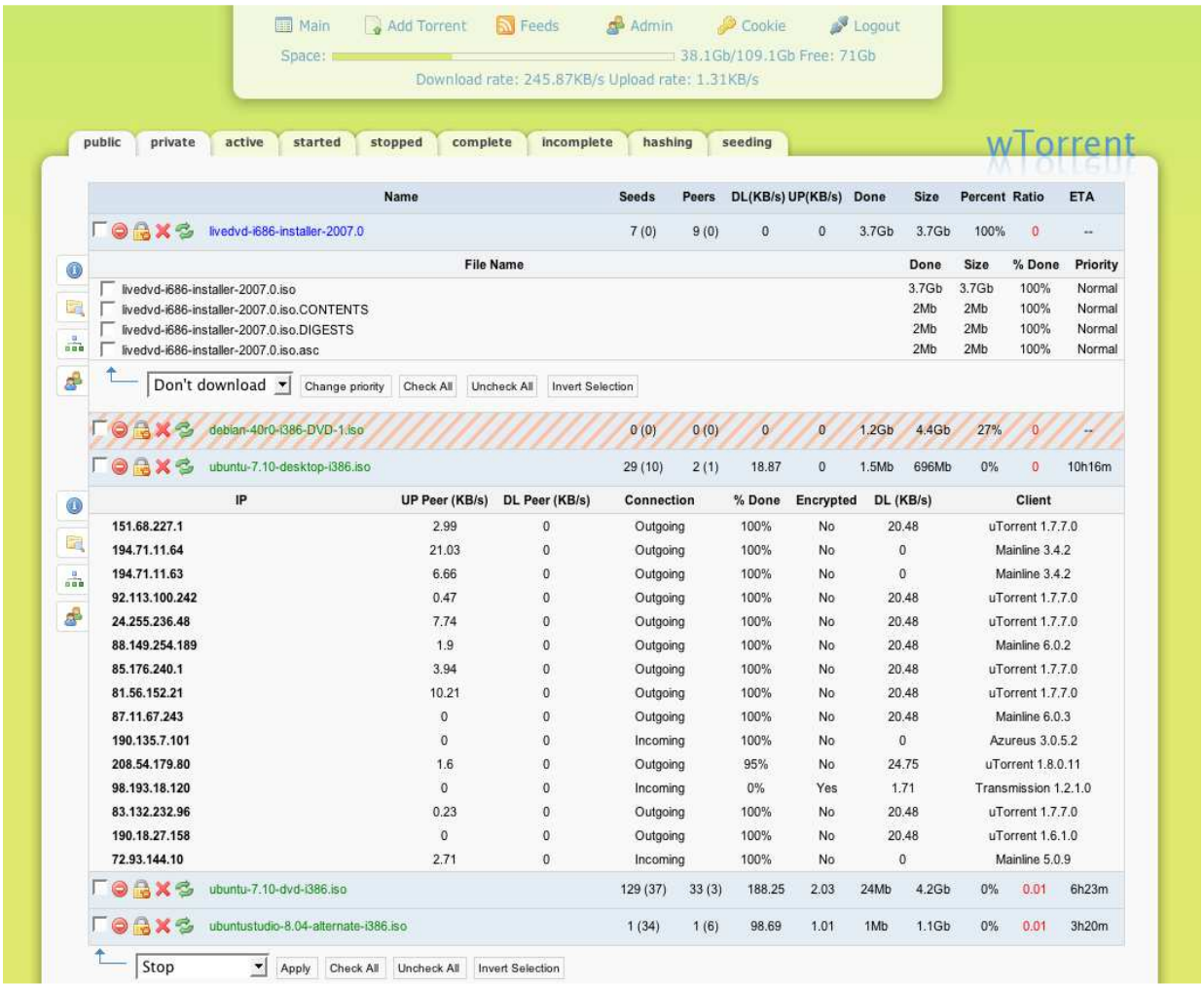

Figure 7: Interface de wTorrent

D'autre part, *wTorrent* fournit une panoplie d'outils très large pour piloter rTorrent. En plus des fonctions traditionnelles, telles que « start », « pause », « stop », *wTorrent* donne des informations sur les clients sources et les clients dont l'utilisateur est la source. L'interface propose également de faire le tri entre les fichiers en cours de téléchargement, les fichiers en pause, et ceux qui sont complets.

Enfin *wTorrent* offre la possibilité de charger des fichiers .torrent à partir d'une URL et de charger des fichiers .torrent disponibles sur l'ordinateur de l'utilisateur par l'intermédiaire de l'interface suivante :

Figure 8: Chargement d'un fichier torrent

ii. XMLRPC

rTorrent gère le protocole XML-RPC . *wTorrent* se base alors sur ce protocole pour communiquer avec rTorrent.

XML-RPC est un protocole RPC (Remote procedure call), une spécification simple et un ensemble de codes qui permettent à des processus s'exécutant dans des environnements différents de faire des appels de méthodes à travers un réseau.

Cependant *wTorrent* et *rTorrent* ne peuvent pas communiquer directement. Il est nécessaire d'installer une passerelle SCGI entre les deux.

Dans notre projet, le serveur Web *Lighttpd* a été installé pour servir de passerelle.

iii. Smarty

wTorrent est programmée en php en utilisant des templates *Smarty*. Il s'agit d'un moteur de templates pour PHP. Il est rapide et permet la gestion des caches. La gestion du cache permet d'accélérer les traitements (la page n'est pas systématiquement recalculée).

L'intérêt principal de *Smarty* réside dans la séparation du contenu et de la forme. Ceci permet de gagner du temps lors de la conception d'un projet et d'aboutir à un code uniforme au sein d'une équipe de développeurs.

Ainsi Smarty est à PHP, ce que la synthèse de haut niveau est au VHDL.

iv. Prérequis

Pour pouvoir utiliser *wTorrent*, il est nécessaire d'installer les applications suivantes sur la borne :

- **Lighttpd** : est un serveur web rapide dont l'empreinte mémoire est faible. Il intègre le support fastCGI et SCGI. FastCGI est une technologie permettant la communication entre un serveur HTTP et un logiciel externe. A l'inverse de CGI, FastCGI utilise un ensemble fini de processus pour traiter les requêtes. Ceci permet un traitement plus rapide des requêtes. Dans notre cas FastCGI est utilisé entre *Lighttpd* et *PHP*. SCGI est similaire à FastCGI mais son implémentation est plus simple. Dans notre cas, SCGI est utilisé entre le rTorrent et le serveur *Lighttpd* pour faire transiter les commandes xmlrpc.
- **Xmlrpc-c** permet la communication entre wTorrent et rtorrent.
- **PHP** doit être compilé avec les extensions php-xmlrpc, php-pdo, sqlite et curl.

v. Modifications

Même si l'interface *wTorrent* est très complète, il manque quelques fonctionnalités pour pouvoir correspondre au cahier des charges. Ainsi nous avons ajouté ces fonctionnalités en utilisant les templates *Smarty*.

1. *Liste des fichiers .torrent disponibles sur le Media Server.*

Il s'agit d'intégrer dans *wTorrent* une liste des fichiers .torrent disponibles sur le Media Server. L'idée est de réaliser une interface pour que l'utilisateur puisse en un seul clic de souris choisir et télécharger le fichier qu'il désire. L'utilisateur choisit l'un des fichiers disponibles et le routeur Wifi s'occupe de télécharger le .torrent correspondant et de lancer le téléchargement du media dans *rTorrent*.

Sur le Media Server, le tracker *XBNBT* fournit la liste des fichiers .torrent disponibles et donne des liens hypertextes permettant de télécharger ces .torrent.

La solution choisie est alors la suivante. Lorsque l'utilisateur demande à voir la liste des .torrent disponibles, *wTorrent* récupère la liste donnée par *XBNBT* modifie les liens hypertextes pour les adapter *à wTorrent* puis les affiche à l'utilisateur. Lorsque l'utilisateur sélectionne un media en cliquant sur le lien désiré, la requête est envoyée à *wTorrent* qui s'occupe alors de télécharger les fichiers requis.

Le séquencement est résumé dans la figure suivante :

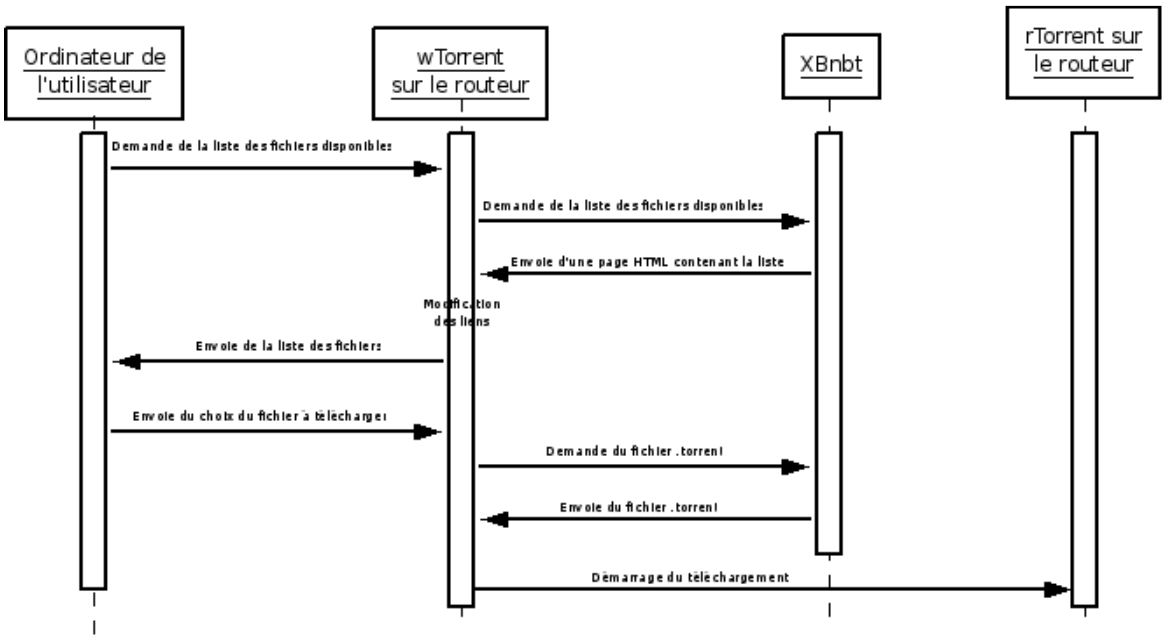
Ordinateur de l'utilisateur
wTorrent sur le routeur
XBnbt
rTorrent sur le routeur
Demande de la liste des fichiers disponibles
Demande de la liste des fichiers disponibles
Envoie d'une page HTML contenant la liste
Modification des liens
Envoie de la liste des fichiers
Envoie du choix du fichier à télécharger
Demande du fichier .torrent
Envoie du fichier .torrent
Démarrage du téléchargement

Figure 9: Séquencement des opérations suite à une requête utilisateur

Pour réaliser cette nouvelle fonctionnalité, il a fallu modifier les sources de *wTorrent*. Les sources sont données en annexes.

- **FileList.cls.php :** contient la nouvelle classe définissant les méthodes nécessaires. Il est situé dans wt/cls.
- **fileList.tpl.php** : effectue le téléchargement de la page donnée par XBNBT. Il modifie ensuite les liens en remplaçant ceux donnés par *XBNBT* par des liens adaptés à *wTorrent*. Par exemple, le lien files/torrents/Nemo.torrent, devient «./index.php?cls=AddT&torrenturl=http://adresseMediaServer:6969/files/torrents/Nemo.torrent» FileList.tpl.php est situé dans wt/tpl. Par ailleurs, *XBNBT* fournit un champ de recherche de fichiers. Les données de ce champ sont envoyées par la méthode Post à *XBNBT*. Pour que *wTorrent* puisse traiter ces recherches, la demande de recherche est désormais envoyée à *wTorrent*.
- **Fichiers de langues** : il est nécessaire d'ajouter une correspondance entre le nom de la classe et la langue choisie. Les fichiers de langues sont situés dans wt/lang.
- **Classe rTorrent :** pour que la nouvelle page puisse apparaître dans le menu, il a fallu modifier la classe rTorrent en ajoutant la référence à FileList.
- **Icône :** pour qu'une icône apparaisse à coté du nom de la page dans le menu, il a été nécessaire d'ajouter un fichier image fileList.png dans le dossier wt/img/menu.
- **Xbnbt.css :** la page html envoyée par *XBNBT* utilise des classes css. Il a donc fallu ajouter ce fichier dans wt/css de *wTorrent*. Nous avons retiré la description de la couleur du fond de ce fichier pour qu'il n'y ait pas de conflit avec la couleur du fond de wTorrent. Ensuite nous avons ajouté la ligne : <link rel="xbnbt" href="{$DIR_CSS}xbnbt.css" type="text/css" /> dans le fichier wt/tpl/index.tpl.php.

La capture d'écran suivante montre le résultat obtenu :

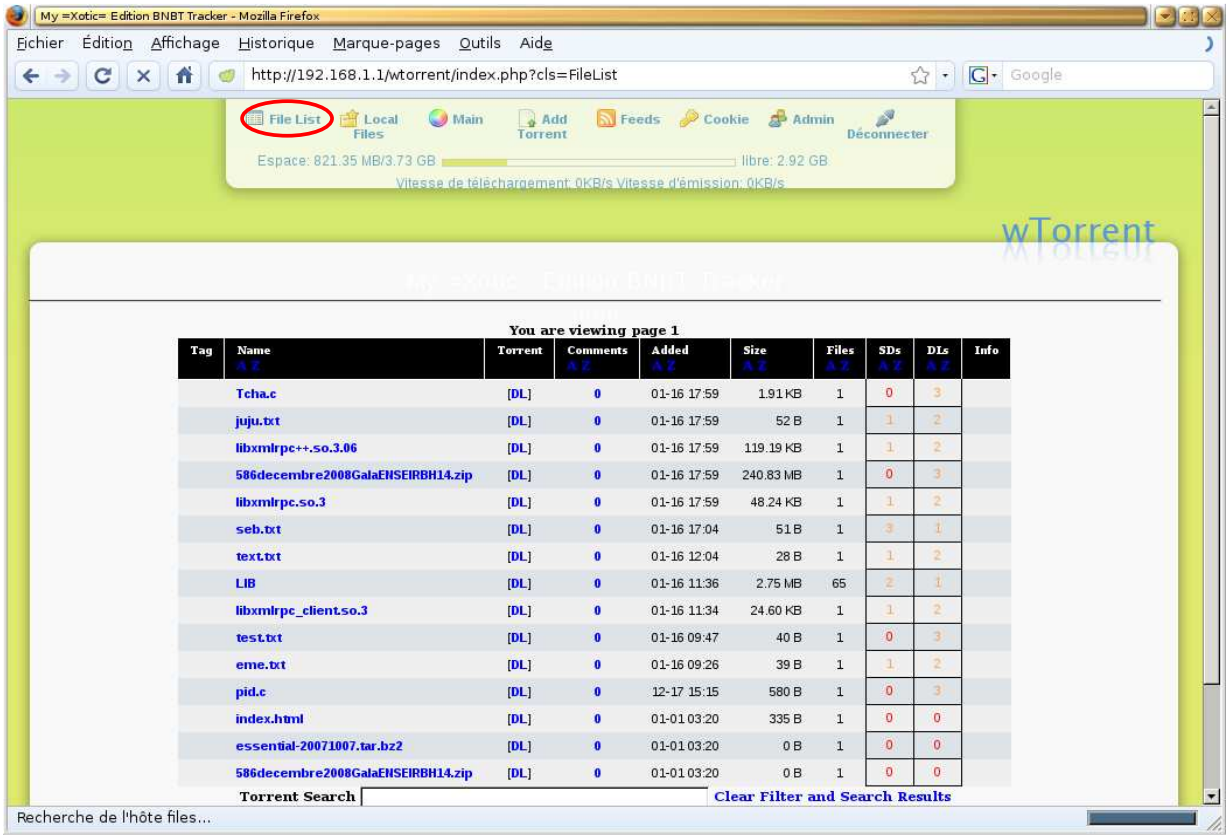
My =Xotic= Edition BNBT Tracker - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
http://192.168.1.1/wtorrent/index.php?cls=FileList
File List
Local Files
Main
Add Torrent
Feeds
Cookie
Admin
Déconnecter
Espace: 821.35 MB/3.73 GB
libre: 2.92 GB
Vitesse de téléchargement: 0KB/s Vitesse d'émission: 0KB/s
wTorrent
You are viewing page 1
Tag
Name
Torrent
Comments
Added
Size
Files
SDs
DLs
Info
Tcha.c [DL] 0 01-16 17:59 1.91 KB 1 0 3
juju.txt [DL] 0 01-16 17:59 52 B 1 1 2
libxmlrpc++.so.3.06 [DL] 0 01-16 17:59 119.19 KB 1 1 2
586decembre2008GalaENSEIRBH14.zip [DL] 0 01-16 17:59 240.83 MB 1 0 3
libxmlrpc.so.3 [DL] 0 01-16 17:59 48.24 KB 1 1 2
seb.txt [DL] 0 01-16 17:04 51 B 1 3 1
text.txt [DL] 0 01-16 12:04 28 B 1 1 2
LIB [DL] 0 01-16 11:36 2.75 MB 65 2 1
libxmlrpc_client.so.3 [DL] 0 01-16 11:34 24.60 KB 1 1 2
test.txt [DL] 0 01-16 09:47 40 B 1 0 3
eme.txt [DL] 0 01-16 09:26 39 B 1 1 2
pid.c [DL] 0 12-17 15:15 580 B 1 0 3
index.html [DL] 0 01-01 03:20 335 B 1 0 0
essential-20071007.tar.bz2 [DL] 0 01-01 03:20 0 B 1 0 0
586decembre2008GalaENSEIRBH14.zip [DL] 0 01-01 03:20 0 B 1 0 0
Torrent Search
Clear Filter and Search Results
Recherche de l'hôte files...

Figure 10: Listes des fichiers disponibles sur le tracker

En cliquant sur « DL » correspondant au fichier, le téléchargement se lance tout seul.

Cependant des problèmes sont visibles lorsque l'utilisateur demande les informations du fichier en cliquant sur le nom du fichier et lorsque l'utilisateur demande à voir les commentaires. La page affichée est vide. Comme le temps manquait, nous n'avons pas pu corriger ce problème. En revanche, la recherche de fichiers fonctionne.

2. *Liste des fichiers disponibles sur le routeur*

Les fichiers sont téléchargés sur la clé USB du routeur. Pour que l'utilisateur puisse récupérer les fichiers qu'il a téléchargés, il était nécessaire de développer une interface listant tous les fichiers disponibles sur la borne et proposant des liens pour les télécharger. Nous avons donc adapté *wTorrent* à cette nouvelle spécification :

- **LocalFiles.cls.php :** contient la nouvelle classe définissant les méthodes nécessaires. Elle contient notamment une méthode listant les fichiers disponibles sur la clé USB. Il est situé dans wt/cls.
- **localFiles.tpl.php** : appelle la méthode de listing et l'ajoute au contenu php de la page.
- **Fichiers de langues** : il est nécessaire d'ajouter une correspondance entre le nom de la classe et la langue choisie. Les fichiers de langues sont situés dans wt/lang.
- **Classe rTorrent :** pour que la nouvelle page puisse apparaître dans le menu, il a fallu modifier la classe rTorrent en ajoutant la référence à LocalFiles.

- **Icône :** pour qu'une icône apparaisse à coté du nom de la page dans le menu, il a été nécessaire d'ajouter un fichier image localFiles.png dans le dossier wt/img/menu.

Le résultat est le suivant :

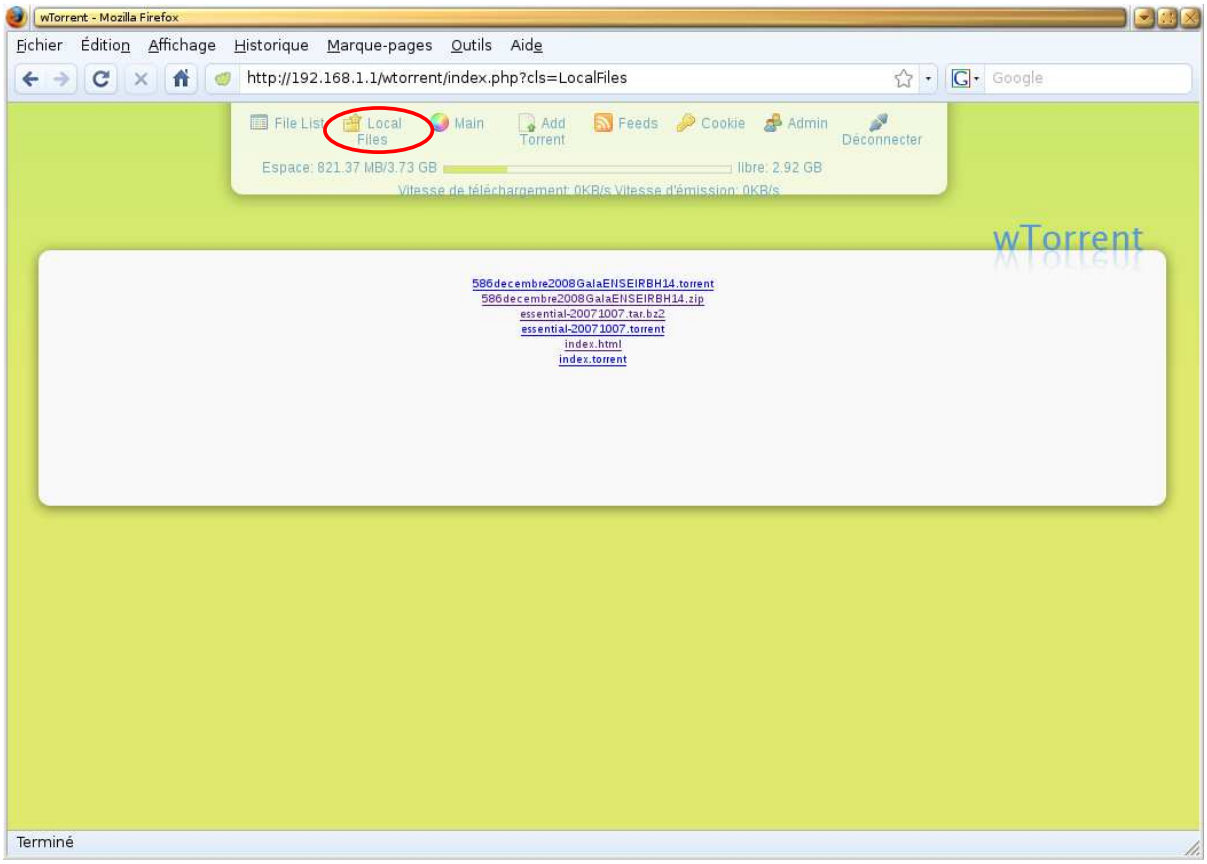

Figure 11 : Liste des fichiers locaux

En cliquant sur le fichier désiré, l'utilisateur télécharge sur sa machine le media.

En revanche, nous nous sommes aperçus qu'il y avait des problèmes de téléchargement avec certains fichiers. Nous n'avons pas eu le temps de corriger ce problème.

vi. Utilisation

Pour pouvoir accéder à l'interface wTorrent, l'utilisateur à simplement à entrer l'adresse http://192.168.1.1/wtorrent dans son navigateur web.

7. La compilation croisée

Pour pouvoir fournir les fonctionnalités requises par le cahier des charges, nous avons compilé un certain nombre d'applications spécialement pour le processeur MIPSel. Plusieurs options sont nécessaires à la compilation de chacune des sources. Pour éviter d'avoir à réécrire ces options à chaque compilation, nous avons écrit des scripts. Ces scripts de compilation sont donnés en annexes. Chacun des scripts contient la même entête :

- Une redéfinition du PATH pour y ajouter le chemin vers les compilateurs croisés.
- Une redéfinition de LDFLAGS pour indiquer au linker où se trouvent les librairies de Kamikaze
- Une redéfinition de CXXFLAGS et CPPFLAGS pour indiquer au compilateur mipsel-linux-g++ où sont situées les « includes », et pour spécifier l'optimisation -Os (optimisation en taille)
- Une redéfinition de LIBS et de STUFF_LIBS pour spécifier aux compilateurs la localisation des librairies

Dans chacune des sources compilées figure un fichier « ./configure » permettant de configurer l'environnement de compilation. Pour chacune des applications, « ./configure » est lancé avec deux options :

- --host=mipsel-linux : préfixe nécessaire pour appeler le compilateur croisé (par exemple : mipsel-linux-g++)
- --prefix=DOSSIER_MIPSEL : indique le dossier dans lequel seront installés les programmes et les librairies compilés.

D'autres options ont été spécifiées en fonction de l'application compilée. Ces divergences sont expliquées dans la suite.

Enfin chacun des scripts se termine par la commande « make » et « make install » qui permettent respectivement de compiler les sources et d'installer les binaires créés dans DOSSIER_MIPSEL.

La partie suivante va répertorier tous les programmes ou bibliothèques qui ont dû être « cross compilés » car nécessaires au fonctionnement du réseau.

i. Libsigc++

La libsigc++ implémente les appels systèmes nécessaires à l'exécution de code C++. Comme libtorrent et rTorrent sont écrits en langage C++, cette bibliothèque est nécessaire. La version utilisée est libsigc++-2.0.18 disponible à l'adresse http://libsigc.sourceforge.net

ii. Libtorrent

La libtorrent est nécessaire à l'exécution de rTorrent est n'est pas disponible dans les dépôts de Kamikaze, c'est pourquoi nous l'avons compilée. La dernière version stable en date était libtorrent-0.12.3 disponible à l'adresse : http://libtorrent.rakshasa.no

Dans le cas de cette bibliothèque l'option --enable-debug=no a été ajoutée à la commande « ./configure » afin de retirer les options de debugage mises par défaut. Ceci permet de réduire la taille des binaires créés.

iii. Xmlrpc

Pour pouvoir être contrôlée à distance, rTorrent a besoin de la bibliothèque Xmlrpc.

La version utilisée pour le projet est la version 1.11. Elle est disponible à l'adresse suivante : http://xmlrpc-c.sourceforge.net/

Au niveau de la compilation croisée, rien de plus n'est nécessaire que le script de base décrit en introduction.

iv. rTtorrent

rTorrent n'est pas disponible dans les dépôts de Kamikaze, c'est pourquoi nous l'avons compilé. La version utilisée est la 8.2, à savoir la dernière version stable en date. Il est disponible à l'adresse suivante : http://libtorrent.rakshasa.no

Options de compilation

Pour compiler rTorrent, quelques options ont été ajoutées aux variables d'environnement :

- Les options -lncurses, -lsigc-2.0, -lcurl, et -ltorrent ont été ajoutées aux variables LIBS, STUFF_LIBS et CFLAGS.
- Les chemins vers les includes de la bibliothèque libsigc++ ont été ajoutés aux variables STUFF_CXXFLAGS et STUFF_CFLAGS

Des options ont également été ajoutées au « ./configure ». Il s'agit de :

- --oldincludedir=[DOSSIER] permet de spécifier la localisation des includes
- --with-xmlrpc-c permet d'ajouter le support xmlrpc à rTorrent
- --enable-debug=no permet de retirer toutes les options de debugage et ainsi d'avoir un binaire plus léger.

Modification de cbase_type.h

Il n'a pas été possible de compiler rTorrent directement. Il a fallu modifier le fichier ctype_base.h situé dans le dossier *kamikaze/staging_dir/toolchain-mipsel_gcc4.1.2/include/c++/4.1.2/mipsel-linux-uclibc/*bits pour le rendre compatible avec rTorrent : le type « __ctype_mask_t » génère des erreurs de compilation. Il a fallu le remplacer par le type « int ».

Compilation manuelle

Le script de compilation croisée exécute la commande *./configure*. Il permet de générer des Makefiles adaptés à l'environnement de compilation. Cependant il met par défaut le dossier *-I/usr/include* aux options de compilation. Ceci gène la compilation de certaines sources du projet rTorrent car ces

includes ne correspondent pas à ceux de Kamikaze. C'est pourquoi, il est nécessaire de compiler manuellement certains fichiers sources en retirant cette option. Ceci concerne les sources :

- src/core/manager.cc
- src/command_network.cc
- src/command_peer.cc

Le script de compilation que nous avons écrit exécute la compilation de ces fichiers avant d'effectuer la commande « make ».

Link

La dernière instruction de compilation effectuée par « make » consiste à linker tous les fichiers .o compilés.

Pour effectuer cette tâche sur le projet rTorrent, « make » utilise la suite libtool. Cependant libtool cherche le fichier /usr/local/lib/libsigc-2.0.0.la alors qu'il devrait le chercher dans les libraries de Kamikaze. Ceci génère donc une erreur. C'est pourquoi il faut effectuer le link manuellement.

Le script de compilation que nous avons écrit effectue le link après compilation de tous les fichiers sources.

v. Libxml

Pour que php puisse gérer les fonctions xml, il a été nécessaire de compiler la *libxml*. Pour pouvoir être compilée, il a été nécessaire d'ajouter à la variable CFLAGS l'option -lcurl qui permet d'ajouter les fonctions curl. La version que nous avons utilisée est libxml2-2.7.2 disponible à l'adresse : http://xmlsoft.org

Par défaut les options de debugage sont activées. Nous les avons désactivées pour que les binaires utilisent moins de place en mémoire. C'est pourquoi dans le script de compilation, « ./configure » comporte l'option : *--with-debug=no*

vi. PHP

Nous avons choisi d'installer PHP pour pouvoir répondre au mieux aux objectifs. Cependant, PHP n'est pas disponible dans les dépôts de Kamikaze, c'est pourquoi nous l'avons recompilé. La version utilisée est php-5.2.8.

Pour pouvoir compiler PHP, il a fallu ajouter un certain nombre d'options à « ./configure » dans le script de compilation :

- --enable-sockets : cette option permet d'ajouter le support des sockets à PHP. Il peut ainsi dialoguer avec d'autres serveurs.
- --enable-fastcgi : cette option ajoute le support fastcgi. Le protocole Fastcgi est utilisé entre Lighttpd et PHP pour communiquer. Ainsi Lighttpd envoie les pages php à php-cgi pour les interpréter. Ce dernier retourne les pages html à Lighttpd pour qu'il puisse les envoyer au client.
- --without-iconv : *iconv* a été désactivé car il n'est pas utile dans notre projet. *Iconv* est une

library permettant de convertir un fichier dans un autre jeu de caractères. En désactivant ce module un gain de place est réalisé.

- --without-pear : *PEAR* a été désactivé car il n'est pas utile dans notre projet. *PEAR* fournit une bibliothèque de scripts php. PEAR permet de promouvoir le « re-use ». Elle est utile dans le cas de développement de grands projets php.
- --with-libxml-dir=DOSSIER : permet le support de xml. DOSSIER correspond au chemin de la librairy compilée précédemment.
- --with-curl=DOSSIER : permet la prise en charge de curl. Curl offre des fonctions pour transférer des fichiers.

Cependant php ne permet pas officiellement la compilation croisée. Pour contourner, ce problème il faut lancer une première fois « ./configure ». Suite à cette exécution un fichier « config.cache » est créé. Il faut alors éditer ce fichier en remplaçant :

- *ar* par *mipsel-linux-ar*
- *gcc* par *mipsel-linux gcc*
- *ranlib* par *mipsel-linux-ranlib*
- *strip* par *mipsel-linux-strip*

La compilation peut désormais être effectuée en lançant le script de compilation que nous avons écrit.

vii. Lighttpd

Lighttpd est disponible dans les dépots de Kamikaze, mais cette version n'est pas à jour c'est pourquoi nous l'avons recompilée. La version utilisée est lighttpd-1.4.20, disponible à l'adresse http:// www.lighttpd.net .

Par rapport au script de compilation de base, des options au ./configure ont été ajoutées :

- --without-zlib
- --without-bzip2

Comme nous n'utilisons pas ces fonctionnalités dans le projet, nous les avons retirées afin de gagner en taille mémoire.

viii. Ncftp

Les scripts mettant à jour le dépôt de torrents sur le serveur de Media utilisent l'utilitaire *Ncftp.* Comme *Ncftp* n'est pas disponible dans les dépôts de Kamikaze, nous l'avons compilé. La version que nous avons utilisée est ncftp-3.2.2 disponible sur http://www.ncftp.com.

Le script de compilation croisée de base que nous avons écrit suffit.

Cependant, tout comme pour PHP, les sources Ncftp ne prennent pas en charge la compilation croisée. Pour palier ce problème il faut d'abord lancer « ./configure ». Suite à cette exécution un fichier config.cache est créé. Il faut alors éditer ce fichier en remplaçant *gcc* par *mipsel-linux-gcc*. Il faut également écrire « no » au lieu de « yes » au niveau des variables *ac_cv_funct_gnu_get_libc_release* et *ac_cv_funct_gnu_get_libc_version*. Cette dernière modification est dûe au fait que nous n'utilisons

pas *libc* sur la borne mais *ulibc*. *Ulibc* est une version de *libc* destinée à l'embarqué.

Une fois ces modifications effectuées, le script de compilation peut être lancé. Parmi tous les exécutables créés (*ncftp, ncftpls, ncftpput , ncftpget, ncftpbookmarks....*), nous n'avons copié sur la borne que ceux dont nous avions besoin, c'est à dire *ncftpput* et *ncftpls*.

Conclusion

Au terme de ce projet, nous avons réussi à installer un client BitTorrent sur un routeur ASUS. De plus celui-ci possède un espace de stockage amovible sur port USB, permettant ainsi de varier la taille de celui-ci en fonction des besoins.

De même, le serveur de médias a été réalisé et peut être installé sur un simple ordinateur de bureau.

Cependant, quelques fonctionnalités n'ont pas pu être implémentées. Le routeur initialise à son démarrage le serveur web qui permet de contrôler l'ajout de téléchargements, mais le client BitTorrent ne se lance pas automatiquement. Son démarrage doit se faire manuellement. De même, il faut initialiser manuellement rTorrent sur le serveur, cette tâche n'a pu être automatisée.

De plus, il manque une interface pour charger des fichiers depuis l'ordinateur de l'utilisateur sur le routeur. En effet pour le moment, cette manipulation n'est possible que par une connexion ssh. Nous pourrions installer un serveur FTP pour faciliter la tâche à l'utilisateur.

Dans ce projet, nous avons découvert et utilisé un système d'exploitation optimisé pour des routeurs wifi : OpenWRT. Le projet nous a permis de découvrir l'utilité d'un tel système dans les routeurs wifi. De plus, nous nous sommes familiarisés avec la création de trackers BitTorrent.

Annexes

Ces annexes ont pour but de présenter un court mode d'emploi du fonctionnement des différents outils composant le réseau Peer to Peer. Les codes sources du projet sont également données dans ces annexes.

I. Mode d'emploi

1. *Lancement du Media Server (ordinateur)*

- Démarrer l'ordinateur
- Se logger sous l'utilisateur « loutre » avec comme mot de passe « loutre »
- Ouvrir une console et lancer la commande « rTorrent »

Le Media Server est désormais opérationnel.

2. *Lancement du routeur*

- Brancher la clé USB sur l'un des ports
- Allumer le routeur
- A partir d'un ordinateur relié au réseau, lancer la commande : ssh root@192.168.1.1
- Un mot de passe est demandé. Saisir : « root »
- Saisir la commande : « rtorrentd start »
- Saisir la commande « exit »

Le routeur est désormais opérationnel.

3. *Lancer un téléchargement*

- Depuis l'ordinateur de l'utilisateur relié au réseau, ouvrir un navigateur web.
- Se rendre à l'adresse : http://192.168.1.1/wtorrent
- Cliquer sur l'onglet « File List »
- Sélectionner un fichier en cliquant sur « DL »

4. Rapatrier un fichier sur l'ordinateur

- Depuis l'ordinateur de l'utilisateur relié au réseau, ouvrir un navigateur web.
- Se rendre à l'adresse : http://192.168.1.1/wtorrent
- Cliquer sur l'onglet « Local Files »
- Sélectionner un fichier en cliquant sur le lien correspondant

5. Administrer le Media Server

i. Ajout de fichiers

Pour ajouter un fichier sur le Media Server, il suffit de le copier dans le dossier /usr/local/bin/xbnbt/files/torrents.

ii. Administrer XBNBT

- Se rendre à l'adresse http://192.168.1.112:6969
- Cliquer sur le lien « login »
- Le nom d'utilisateur est « loutre » et le mot de passe est « loutre »
- Effectuer les modifications nécessaires

II. Sources

1. *Source ajoutées à wTorrent*

i. Liste des fichiers disponibles sur le Media Server

wt/cls/FileList.cls.php

```
<?php
/*
This file is part of wTorrent.

wTorrent is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

wTorrent is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
class FileList extends rtorrent
{
  public function construct()
  {
    if(!$this->setClient())
    {
      return false;
    }

  }

  /*What is the asked page ?*/
public function getServerPage()
{
return $_REQUEST['server_page'];
}

/*Search item*/
public function getSearch()
{
return $_REQUEST['search'];
}

}
?>
```

wt/cls/fileList.tpl.php

```
  <div style="width: 100%; height: 100%;">

  <input type="hidden" name="MAX_FILE_SIZE" value="100000000" />

 {assign var='search' value=$web->getSearch()}

 {assign var='pagehtml' value=$web->getServerPage()}

{if empty($pagehtml)}
 {assign var='pagehtml' value="index.html?search=$search"}
{else}
 {assign var='pagehtml' value="$pagehtml&search=$search"}
{/if}

{fetch file="http://$BLACKBOX/$pagehtml" assign='dl_page'}


/*Remplacement des liens*/
{assign var='dl_page' value=$dl_page|regex_replace:"/<!--.*>/":""}
{assign var='dl_page'
value=$dl_page|regex_replace:"/\/file/":"http://$torrent_server/file"}

{assign var='dl_page'
value=$dl_page|regex_replace:"/\/index\.html/":"./index.php?cls=FileList&se
rver_page=index.html"}

{assign var='dl_page'
value=$dl_page|regex_replace:"/\/stats\.html/":"./index.php?cls=FileList&se
rver_page=stats.html"}
{assign var='dl_page'
value=$dl_page|regex_replace:"/\/login\.html/":"./index.php?cls=FileList&se
rver_page=login.html"}
{assign var='dl_page'
value=$dl_page|regex_replace:"/\/comments\.html/":"./index.php?cls=FileList
&server_page=comments.html"}
{assign var='dl_page'
value=$dl_page|regex_replace:"/files\/torrents/":"./index.php?cls=AddT&torr
enturl=http://$torrent_server/files/torrents"}


/*On enlève les menus login, my torrents, upload, admin, pour ne laisser
que le principal*/
{assign var='dl_page' value=$dl_page|regex_replace:"/.*ogin.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*ign.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*My Torrents.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*pload.*/":""}

{assign var='dl_page' value=$dl_page|regex_replace:"/.*info\.html.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*Tracker Info.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*dmin.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*sers.*/":""}
```

```
{assign var='dl_page' value=$dl_page|regex_replace:"/.*LINK
TEMPLATE.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*clickable text
goes.*/":""}
{assign var='dl_page' value=$dl_page|regex_replace:"/.*END OF
TEMPLATE.*/":""}


/*Affichage de la page*/
{$dl_page}

  </div>
```

ii. Affichage des fichiers locaux sur le routeur

wt/cls/Localfiles.cls.php

```
<?php
/*
This file is part of wTorrent.

wTorrent is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

wTorrent is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
class LocalFiles extends rtorrent
{
  public function construct()
  {
    if(!$this->setClient())
    {
      return false;
    }

  }

/*List files which are the router*/
  public function listDir()
{

  $mydir = "/mnt/cle/wtorrent/data";
  if ($dir = @opendir($mydir))
```

```
  {
    while (($file = readdir($dir)) !== false)
    {
      if($file != ".." && $file != ".")
      {
        $filelist[] = $file;
      }
    }
    closedir($dir);
  if(sizeof($filelist) != '0') {
       sort($filelist);
    }

    for($i=0;$i<sizeof($filelist);$i++)
    {
        echo '<a href="data/'.$filelist[$i].'"> '.$filelist[$i].' </a>
<br>';
    }
  }
}
}
?>
```

wt/cls/localfiles.tpl.php

```
  <div style="width: 100%; height: 180px;">
  <input type="hidden" name="MAX_FILE_SIZE" value="100000000" />

/*Appel de la méthode de listing*/
{$web->listDir()}
  </div>
```

2. Scripts de compilation croisée

i. Compilation croisée de xmlrpc-c

```
# xmlrpc-c

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
#export CFLAGS="-Os -lncurses -lsigc-2.0 -lcurl -ltorrent -
I/home/seb/Projet_Torrent/mips/include"
CFLAGS="-Os -lcurl -I/home/seb/Projet_Torrent/mips/include"
export LIBS="-lncurses  -lsigc-2.0 -L/home/seb/Projet_Torrent/mips/lib -
lcurl -ltorrent"
export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib -lsigc-2.0 -lcurl -
ltorrent"
```

```
./configure  --disable-cplusplus --host=mipsel-linux --build=i486-linux-
gnu --disable-curl-client --prefix=/home/seb/Projet_Torrent/mips || exit

cd lib/expat/gennmtab

#Compilation du fichier gennmtab qui sera utilisé lors de la compilation
gcc -c gennmtab.c -o gennmtab.o -DNDEBUG -Wall -Wundef -Wimplicit -W -
Winline -Wundef -Wmissing-declarations -Wstrict-prototypes -Wmissing-
prototypes -fno-common -Os    -I/home/seb/Projet_Torrent/rtorrent/xmlrpc-c-
1.12.00/xmlrpc-c-1.11.00 -I/home/seb/Projet_Torrent/rtorrent/xmlrpc-c-
1.12.00/xmlrpc-c-1.11.00/lib/util/include && gcc -o gennmtab  gennmtab.o
cd ../../../

make && make install
```

ii. Compilation croisée de rTorrent

```
#rTorrent

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH
export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/include  "
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CFLAGS="-Os -lncurses -lsigc-2.0 -lcurl -ltorrent -
I/home/seb/Projet_Torrent/mips/include"
export LIBS="-lncurses  -lsigc-2.0 -L/home/seb/Projet_Torrent/mips/lib -
lcurl -ltorrent"
export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib -lsigc-2.0 -lcurl -
ltorrent"
export STUFF_CFLAGS="-
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-2.0
-I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include"
export STUFF_CXXFLAGS="-
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-2.0
-I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include"



./configure  --host=mipsel-linux --
oldincludedir=/home/seb/Projet_Torrent/mips/include --with-xmlrpc-c --
enable-debug=no --prefix=/home/seb/Projet_Torrent/mips

#Compilation manuelle des fichiers qui ne supportent pas l'optioin -
I/usr/local/
cd src/core
```

```
mipsel-linux-g++ -DHAVE_CONFIG_H -I. -I../.. -I. -I./.. -I../..  -Os -
I/home/seb/Projet_Torrent/mips/include  -Os -
I/home/seb/Projet_Torrent/mips/include -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/include   -DNDEBUG -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-
2.0 -I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include  -MT manager.o -
MD -MP -MF .deps/manager.Tpo -c -o manager.o manager.cc

cd ..
mipsel-linux-g++ -DHAVE_CONFIG_H -I. -I..   -Os -
I/home/seb/Projet_Torrent/mips/include  -Os -
I/home/seb/Projet_Torrent/mips/include -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/include   -DNDEBUG -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-
2.0 -I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include  -
MT command_network.o -MD -MP -MF .deps/command_network.Tpo -c -
o command_network.o command_network.cc

mipsel-linux-g++ -DHAVE_CONFIG_H -I. -I..   -Os -
I/home/seb/Projet_Torrent/mips/include  -Os -
I/home/seb/Projet_Torrent/mips/include -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/include   -DNDEBUG -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-
2.0 -I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include   -
MT command_peer.o -MD -MP -MF .deps/command_peer.Tpo -c -o command_peer.o
command_peer.cc


make

cd src

#Link final
mipsel-linux-g++  -Os -I/home/seb/Projet_Torrent/mips/include -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/include   -DNDEBUG -
I/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchain-
mipsel_gcc4.1.2/inlude  -I/home/seb/Projet_Torrent/mips/include/sigc++-
2.0 -I/home/seb/Projet_Torrent/mips/lib/sigc++-2.0/include -I/usr/include
-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib -
o rtorrent command_download.o command_events.o command_file.o
command_helpers.o command_local.o command_network.o command_peer.o
command_tracker.o command_scheduler.o command_ui.o control.o globals.o
main.o option_parser.o signal_handler.o ../src/ui/libsub_ui.a
../src/core/libsub_core.a ../src/display/libsub_display.a
```

```
../src/input/libsub_input.a ../src/rpc/libsub_rpc.a
../src/utils/libsub_utils.a -lncurses  -lsigc-2.0 -
L/home/seb/Projet_Torrent/mips/lib -lcurl -ltorrent -
L/home/seb/Projet_Torrent/mips/lib -lsigc-2.0 -lcurl -ltorrent -
L/usr/lib  -lxmlrpc_server -lxmlrpc -lxmlrpc_util -lxmlrpc_xmlparse -
lxmlrpc_xmltok

make install
```

iii. Compilation croisée de PHP

```
# PHP

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export OPTIM="-Os"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"

export CFLAGS="-Os -lcurl -I/home/seb/Projet_Torrent/mips/include"
export LIBS="-L/home/seb/Projet_Torrent/mips/lib"
export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"

./configure --target=mipsel-linux --prefix=/home/seb/Projet_Torrent/mips --
enable-sockets --enable-fastcgi --build=mipsel-linux --host=mipsel-linux --
without-iconv --without-pear --with-libxml-
dir=/home/seb/Projet_Torrent/mips --with-
curl=/home/seb/Projet_Torrent/mips --with-curlwrappers || exit

make && make install
```

iv. Compilation croisée de Ncftp

```
#NCFTP

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH
export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export OPTIM="-Os"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CFLAGS="-Os -lcurl -I/home/seb/Projet_Torrent/mips/include"

export LIBS="-L/home/seb/Projet_Torrent/mips/lib"

export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"
```

```
./configure --host=mipsel-linux --prefix=/home/seb/Projet_Torrent/mips ||
exit
make && make install
```

v. Compilation croisée de Lighttpd

```
#Lighttpd

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export OPTIM="-Os"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CFLAGS="-Os -lcurl -I/home/seb/Projet_Torrent/mips/include"
export LIBS="-L/home/seb/Projet_Torrent/mips/lib"
export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"
./configure --without-bzip2 --without-zlib --host=mipsel-linux --
prefix=/home/seb/Projet_Torrent/mips || exit

make && make install
```

vi. Compilation croisée de Libxml

```
#libxml

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"
export OPTIM="-Os"
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export LIBS="-L/home/seb/Projet_Torrent/mips/lib"
export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"

./configure --target=mipsel-linux --prefix=/home/seb/Projet_Torrent/mips --
host=mipsel-linux --with-debug=no || exit

make && make install
```

vii. Compilation croisée de Libtorrent

```
#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH
export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export LIBS="-L/home/seb/Projet_Torrent/mips/lib"
./configure --host=mipsel-linux --enable-debug=no --
prefix=/home/seb/Projet_Torrent/mips || exit
make && make install
```

viii. Compilation croisée de Libsigc++

```
#ligsigc++

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"

export OPTIM="-Os"

export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"

CFLAGS="-Os -lcurl -I/home/seb/Projet_Torrent/mips/include"

export LIBS="-L/home/seb/Projet_Torrent/mips/lib"

export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"

echo $VAR

./configure --host=mipsel-linux --prefix=/home/seb/Projet_Torrent/mips --
with-random=random --without-ssl || exit

make
make install
```

ix. Compilation croisée de Curl

```
#Curl

#!/bin/bash
export PATH=/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/toolchai
n-mipsel_gcc4.1.2/bin:$PATH

export LDFLAGS="-L/home/seb/Projet_Torrent/mips/lib -lcrypto -
L/home/seb/Projet_Torrent/kamikaze_8.09_rc1/staging_dir/mipsel/usr/lib"


export CXXFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"
export CPPFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"

CFLAGS="-Os -I/home/seb/Projet_Torrent/mips/include"

export LIBS="-L/home/seb/Projet_Torrent/mips/lib"

export STUFF_LIBS="-L/home/seb/Projet_Torrent/mips/lib"

echo $VAR

./configure --host=mipsel-linux --prefix=/home/seb/Projet_Torrent/mips --
with-random=random --without-ssl || exit

make && make install
```